WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Tuesday, February 15, 2005

Hide?	Set Name	Query	Hit Count
		DB=USPT,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L7	L1 same (stor$4 near3 request)	5
☐	L6	L1 same ((buffer$4 or queu$4 or fifo) near3 (request or command or instruction))	17
☐	L5	L1 with ((buffer$4 or queu$4 or fifo) near3 (request or command or instruction))	4
☐	L4	L3 with (based or depend$4 or respons$4)	6
☐	L3	L1 with ((buffer$4 or queu$4 or stor$4 or fifo) near3 (request or command or instruction))	29
☐	L2	L1 same ((buffer$4 or queu$4 or stor$4 or fifo) near3 (request or command or instruction))	75
☐	L1	((determin$4 or detect$4 or sens$4) near5 power$4 near3 (status or mode or state or condition))	21107

END OF SEARCH HISTORY

L7: Entry 1 of 5 File: USPT Oct 19, 2004

DOCUMENT-IDENTIFIER: US 6807595 B2
TITLE: Mobile communication device having a prioritized interrupt controller

Detailed Description Text (8):
Additionally, the interrupt controller, upon receiving an interrupt request, determines whether the microprocessor is in a power shut down mode and, if so, whether the interrupt request is of sufficient priority to justify powering up the microprocessor. If so, the interrupt controller forwards appropriate signals to the microprocessor for powering up the microprocessor so that the microprocessor can process and respond the interrupt request. If not, the interrupt controller merely stores the interrupt request pending a subsequent power up operation by the microprocessor. In this manner, improved power savings are achieved by permitting the microprocessor to remain in a power shut down mode for a longer period of time. As can be appreciated, without the interrupt controller, the microprocessor would likely need to power up in response to every interrupt request, regardless of the priority of the request. The many other advantages are achieved by the interrupt controller as well.

Detailed Description Text (11):
The interrupt requests, along with their relative priority levels, are then forwarded to an interrupt notification unit 134 which processes the interrupt requests to determine whether the microprocessor should be immediately notified of the interrupt request of highest priority. If two or more interrupt requests share the highest priority, notification unit 134 employs a round-robin dispatch unit 136 to select one of the interrupt requests of equal priority for processing first. Other techniques or protocols for selecting among interrupt requests of equal priority may alternatively by employed. In any case, the interrupt notification unit identifies a single interrupt request for processing next. All other interrupt requests are stored within a pending interrupt request storage unit 142 for subsequent processing. Under the control of a control unit 138, the notification unit then determines whether the microprocessor is currently in a power shutdown mode. If so, the control unit determines whether the interrupt request is of sufficiently high priority to justify powering up the microprocessor. If so, the control unit activates a microprocessor power up initiation unit 140 which forwards appropriate signals to the microprocessor for powering up the microprocessor via a power control unit 143 of the microprocessor. The power up signals may be forwarded over IRQ or FRQ lines, to be described below, or may be transmitted via dedicated power-up lines, not separately shown. If the selected interrupt request is not authorized to trigger a power up operation of the microprocessor, then the selected interrupt request, and all other interrupt requests, are stored within storage unit 142 for subsequent processing after the microprocessor has eventually been powered up, perhaps following receipt of a subsequent interrupt request of still higher priority. A determination of whether a particular interrupt request is authorized to trigger a power up operation of the microprocessor may be performed, for example, by examining the priority level of the interrupt request. For example, only interrupt requests having a priority level of five or higher may be designated as being of sufficiently high priority to authorize powering up of the microprocessor; whereas interrupt requests having lower priority levels are not authorized to power up the microprocessor. As can be appreciated, though, other methods or protocols may be employed for determining whether a pending interrupt

request should trigger a power up operation of the microprocessor. In the following, it will be assumed that the microprocessor already has been powered up.

Detailed Description Text (23):
Among other functions, priority controller 308 determines whether the selected interrupt request of highest priority is to be immediately stored within an interrupt vector for forwarding to the microprocessor and determines whether an IRQ signal is to be immediately transmitted to the microprocessor. If an interrupt service request is to be forwarded to the microprocessor, the selected interrupt request is identified by setting a corresponding bit within a forty-bit output vector (INT_VECTOR). The priority controller also received FIQ interrupt requests and forwards them immediately to the microprocessor. Additionally, the priority controller determines whether the microprocessor is currently within a power down state and, if the selected interrupt request is of sufficiently high priority to justify powering up the microprocessor, the priority controller forwards the appropriate signals (via power up connection lines not shown in FIG. 4) to the microprocessor (also not shown). To determine whether the highest priority interrupt request should be stored immediately within the interrupt vector and to determine whether an IRQ should be transmitted immediately to the microprocessor, the priority controller exploits IN_STACK and IN_SERVICE registers (illustrates in FIG. 8) to track the status of an interrupt stack of the microprocessor and interrupt service routine processing performed by the microprocessor, respectively. The internal components of the priority controller will be described in greater detail with respect to the remaining figures which should be taken in combination with the descriptions provided above of the overall functionality of the interrupt controller of the invention.

Previous Doc Next Doc Go to Doc#

L5: Entry 1 of 4 File: USPT Aug 3, 2004

DOCUMENT-IDENTIFIER: US 6772355 B2
TITLE: System and method for reducing power consumption in a data processor having a clustered architecture

Detailed Description Text (34):
Power-down controller 250 detects a tight-loop condition in instruction fetch buffer 305 at instruction decode when tight loops can be defined as those fitting within instruction fetch buffer 305--those that fit within instruction fetch buffer 305 are recognized by the jump displacement and buffer sizing. This may advantageously be implemented in hardware.

Detailed Description Text (35):
Power-down controller 250 detects an idle-loop condition by determining whether the instructions in the tight loop in instruction fetch buffer 305 are non-operations (i.e., if all non-operations, then the tight loop may accurately be considered an idle loop). This may advantageously be implemented in hardware.

Detailed Description Text (52):
Power-down controller 250 monitors instruction fetch buffer 305 for the presence of tight loops (decision step 525). Exemplary power-down controller 250 detects a tight-loop condition in instruction execution pipeline 400 by monitoring instruction fetch buffer 305 at instruction decode when tight loops are defined, namely loops fitting within instruction fetch buffer 305 (i.e., recognized by the jump displacement and buffer sizing). This may advantageously be implemented in hardware. In the event that a tight-loop condition is detected ("Y" Branch of Decision Step 525), then power-down controller 250 operates to power down instruction cache 210 (process step 530; e.g., responsive to identifying a tight-loop condition in instruction fetch buffer 305) pending termination of the tight loop.

Detailed Description Text (53):
Power-down controller 250 monitors each tight loop for the presence of idle loops (decision step 535). Exemplary power-down controller 250 detects an idle-loop condition in instruction execution pipeline 400, illustratively by monitoring instruction fetch buffer 305 to determine whether the instructions in the tight loop in instruction fetch buffer 305 are non-operations (i.e., if all non-operations, then the tight loop may accurately be considered an idle loop). This may advantageously be implemented in hardware. In the event that an idle-loop condition is detected ("Y" Branch of Decision Step 535), then power-down controller 250 operates to power down data processor 100 (process step 540), thereby stalling data processor 100 pending an interrupt.

CLAIMS:

2. The data processor as set forth in claim 1 further comprising an instruction fetch buffer, and wherein an identified power-down condition indicates detection of at least one of (i) a non-operation in one of said clusters, (ii) a tight-loop condition in said instruction fetch buffer, and (iii) an idle-loop condition.

15. The processing system as set forth in claim 14 further comprising an

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L4: Entry 2 of 6 File: USPT Aug 6, 2002

DOCUMENT-IDENTIFIER: US 6430687 B1
TITLE: Boot sequence for a network computer including prioritized scheduling of boot code retrieval

CLAIMS:

10. A client computer for use in a computer network, comprising: a power status indicator; and a nonvolatile storage device configured with instructions comprising a boot code sequence, wherein the client computer is operable to execute the boot code sequence in response to a boot event and further wherein the boot code sequence queries the power status indicator and schedules the retrieval of boot code data from a server of the computer network responsive to detecting a power status indicator indicative of a power failure associated with the client computer.

Previous Doc Next Doc Go to Doc#